Ministère de l'Education Nationale

UNIVERSITE MONTPELLIER II
SCIENCES ET TECHNIQUES DU LANGUEDOC

IUP
GENIE MATHEMATIQUE ET INFORMATIQUE

RAPPORT DE STAGE

effectué au

Centre IRD de Montpellier, Laboratoire d'Hydrologie
Du 1er octobre 1998 au 23 janvier 1999

par

Jérome CRESTIN

Directeur de Stage :
Mme NIEL Hélène, Ingénieur de recherche.

LA REGRESSION LINEAIRE FLOUE

Centre National d'Informatique Appliquée de Montpellier
860, rue de Saint Priest - 34090 - MONTPELLIER

Tél. : 04 67 63 13 76 (secrétariat) Fax : 04 67 52 72 63

REMERCIEMENTS:

Je remercie tout d'abord l'IRD (l'Institut de Recherche pour le Développement), l'organisme de recherche en partenariat avec la plupart des pays du sud, de m'avoir accueilli. Je remercie principalement Mme Hélène Niel pour son suivi durant toute la durée de mon stage, ainsi que M. François Delclaux pour son aide dans la compréhension de la logique floue. Je remercie également M. Aubert, qui a contribué au bon déroulement de mon stage en tant que tuteur, ainsi que tout le personnel du Laboratoire d'Hydrologie de l'IRD de Montpellier pour son accueil chaleureux, et plus particulièrement M. Bernard Cappelaere pour son aide dans la découverte du langage Fortran. Et je tiens, pour finir, à remercier M. Lucien Duckstein pour sa conférence et ses explications sur la régression linéaire floue, ainsi que M. Sébastien Roret pour notre collaboration durant nos stages.

La Régression Linéaire Floue:

1. Introduction :

• Présentation de l'IRD :

L'IRD dans le monde

L'IRD (Institut de Recherche pour le Développement), anciennement ORSTOM (Institut Français de Recherche pour le Développement en Coopération) a été créé le 11 octobre 1943 par un acte officiel du centre national de recherche scientifique (CNRS). C'est un organisme public placé aujourd'hui sous la tutelle de la recherche,
La tâche de l'IRD est de contribuer à la connaissance scientifique et au développement des environnements physique, biologique et humain dans les zones intertropicales. Cette tâche s'effectue au moyen de programmes basés sur quatre priorités :

- La compréhension des macro-écosystèmes et la protection de l'environnement.
- La mise en place de pratiques agricoles viables à long terme en régions tropicales fragiles.
- L'environnement et la santé publique.
- L'évolution des milieux et le dynamisme des sociétés et des économies nationales.

Le personnel de recherche est localisé en France et dans les pays partenaires (en Afrique, autour de l'Ocean Indien, en Amérique Latine, en Asie et dans l'Océan Pacifique).

Le centre de Montpellier

C'est le centre en France le plus récent. Il a été inauguré en décembre 1988. L'IRD à Montpellier travaille plus particulièrement sur l'étude du mécanisme des ressources naturelles. Ainsi, les sciences de la vie (biologie végétale, santé humaine) et les sciences de l'environnement(climatologie, hydrologie, ressources en sol, et exploitation des milieux) constituent les principaux domaines abordés par le personnel du centre.

Le centre IRD de Montpellier dispose de 1000 m2 de serres tropicales, de salles de conservations et de cultures de matériels biologiques, une chaîne d'analyse en biochimie, un séquenceur d'ADN, un laboratoire de haute sécurité pour la recherche contre le SIDA, des stations de réceptions de satellites,...

L'hydrologie à Montpellier

L'hydrologie est «la science qui traite des propriétés mécaniques, physiques et chimiques des eaux marines et continentales ».
L'axe scientifique de l'IRD est l'hydrologie quantitative de surface en zone intertropicale qui se donne pour objectif la description des régimes hydroclimatiques à l'échelle de grands ensembles géographiques, et l'étude des mécanismes mis en œuvre dans le cycle de l'eau (précipitation, évaporation, ruissellement, et infiltration).

Le laboratoire assume les fonctions suivantes :

- Constituer et gérer une banque de données pluviométrique et hygrométrique.
- Développer un traitement informatique approprié et performant (statistique, modélisation, représentation cartographique, ...).
- Effectuer des recherches en technologie hydrologique.

Il dispose d'un capital scientifique important :

- Une banque de données recueillie dans le monde inter-tropical depuis cinquante ans.
- Des fonds documentaires en hydrologie tropicale.
- Des logiciels.

Il dispose aussi d'équipements spécifiques :

- Réseaux de stations de travail.
- Stations de réceptions de satellites : Argos et MeteoSat.

• Les objectifs du stage :

L'IRD travaille donc principalement avec des pays en voie de développement. Les moyens mis en place par ces pays ne sont pas toujours suffisants pour disposer en quantité et en qualité des données qui seraient nécessaires au bon déroulement des études hydrologiques. C'est donc dans ce contexte que se situe l'intérêt porté par certains hydrologues sur les techniques de logique floue. Le but de ce stage est de comprendre les fondements de la régression linéaire floue, de la programmer et de la comparer à la méthode de régression statistique classique. Une bibliothèque pré-existante sur le langage flou pourra être utilisée.

2. Présentation du sujet :

• *La régression linéaire statistique.*

En statistique, toute élaboration d'un modèle se décompose en trois parties. Sur un premier jeu de données, le plus important, on émet une calibration d'un modèle avant de le valider sur un deuxième jeu de données, afin par la suite de pouvoir l'utiliser pour obtenir des estimations.

Le but de la régression linéaire statistique est d'établir une relation linéaire entre plusieurs variables dites explicatives X_i et une variable dite expliquée Y_i de la manière suivante(Dagnelie, 1988) :

$$Y_i = \alpha_0 + \sum_{j=1}^{N} \left(\alpha_j . x_{i,j} \right) + D_i \qquad (1)$$

Les D_i sont appelés les résidus de la régression, on détermine l'équation de la régression en minimisant la somme de leur carré. La régression linéaire possède tout de même plusieurs contraintes sur les variables. Les variables explicatives doivent être indépendantes les unes pas rapport aux autres et les résidus doivent vérifier un certain nombre d'hypothèses statistiques telles que la normalité, l'homoscédasticité (variance constante) (Dagnelie, 1988).

Cette relation peut aussi s'écrire de la manière suivante avec pour chaque variable explicative un point de référence représenté par la moyenne(Dagnelie, 1988) :

:

$$Y_i = \alpha_0 + \sum_{j=1}^{N} \left(\alpha_j . \left(x_{i,j} - \overline{x}_j \right) \right) + D_i \qquad (2)$$

Ces formulations désignent : la régression linéaire au sens des moindres carrés.

• Introduction aux ensembles flous et aux nombres flous.

La théorie des ensembles flous a été introduite par Zadeh en 1965. Elle a longtemps concerné un nombre restreint d'adeptes parmi les chercheurs et les enseignants. Utilisée en priorité en mathématique, elle s'est révélée féconde dans de nombreuses applications. Un ensemble flou est un ensemble dont les limites ne sont pas spécifiquement définies, par opposition aux ensembles classiques. Le flou fait correspondre un ensemble à une notion vague du langage (par exemple grand, moyen, petit).

Les ensembles flous :

Un ensemble flou A est défini sur un univers U par une fonction d'appartenance notée μ_A qui retourne une valeur comprise entre zéro et un pour tout élément u de U :

$$0 \leq \mu_A(u) \leq 1$$

Chaque ensemble flou possède plusieurs caractéristiques dont voici les deux principales :

- Le support d'un ensemble flou représente l'ensemble des éléments de U dont le degré d'appartenance est non nul (Tong-Tong, 1985) :

$$Supp(A) = \{u \in U ; \mu_A(u) \neq 0\}$$

- Le support d'un ensemble à un niveau h, note support_h, représente l'ensemble des éléments de U dont le degré d'appartenance est supérieur ou égal à h :

$$Supp_A(h) = \{u \in U ; \mu_A(u) \geq h\}$$

- La hauteur d'un ensemble flou est représentée par la valeur maximale de la fonction d'appartenance sur U (Tong-Tong, 1985) :

$$H(A) = Max(\mu_A(u); u \in U)$$

Les nombres flous :

Les nombres flous sont des ensembles flous particuliers qui vérifient certaines propriétés (Tong-Tong, 1985) :

- U représente l'ensemble des réels.
- La hauteur de l'ensemble est égale à 1.
- La fonction d'appartenance est convexe :

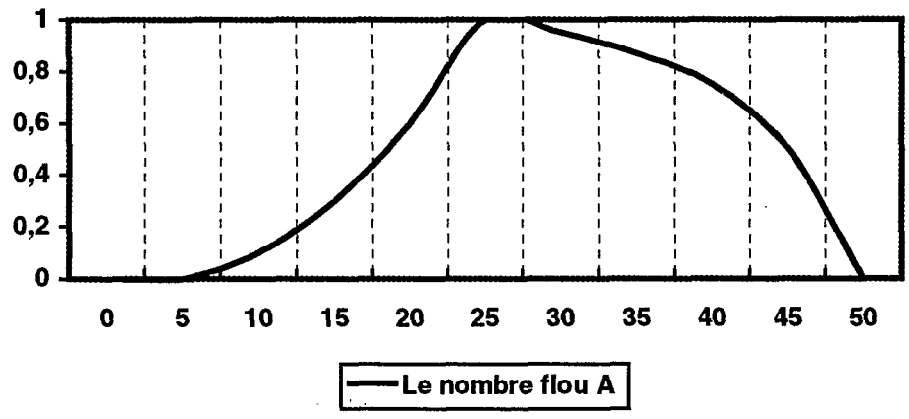
$$\forall a, \forall b, \forall c, \text{ Si } a < b < c \text{ Alors } \mu_A(b) \geq Min(\mu_A(a), \mu_A(c))$$

Ils sont généralement représentés de la manière suivante : (A.Bardossy, 1990)

$$A(\Omega, \alpha, \beta)_{L\text{-}R}$$

- Ω représente le centre de A (L'élément dont la valeur de la fonction d'appartenance est égale à 1). Dans cette formulation des nombres flous, le centre est un point unique, il ne peut pas exister deux éléments distincts dont la valeur de la fonction d'appartenance soit égale à 1.
- α représente le support de A situé à gauche du centre.
- β représente le support de A situé à droite du centre.
- L et R représentent les fonctions d'appartenance respectivement à gauche et à droite du centre.

Voici par exemple la représentation graphique du nombre : $A(25,20,25)_{L\text{-}R}$



(Figure 1)

Les nombres flous L-R au sens de Dubois et Prade :

Ce sont des cas particuliers des nombres flous dits nombres L-R qui se définissent de la manière suivante :

$$L(x) = 1 - x^p \quad \text{et} \quad R(x) = 1 - x^q \tag{3}$$

Avec 'p' et 'q' deux réels généralement compris entre 0 et 1. Lorsque p=q=1 le nombre flou défini est un nombre triangulaire et si p et q sont supérieurs à 1, le nombre flou obtenu est un nombre trapézoïdal.
Cette définition des nombres L-R est la plus utilisée, puisqu'elle reflète parfaitement les courbes curvilignes des nombres L-R, et propose une infinité de courbes tout en étant fort bien utilisable par des outils informatiques.

Le principe d'extension des nombres flous (Duckstein 1998) :

Soit plusieurs nombres flous $A_1 \ldots A_n$ définis respectivement sur $X_1 \ldots X_n$. Et soit une fonction f telle que : $f: X_1,\ldots,X_n \rightarrow Y$
L'image B des nombres flous A par la fonction f sera la fonction d'appartenance suivante :

$$\mu_B(y) = \begin{cases} Max\left[Min\left\{ \mu_{A_1}(x_1),\ldots,\mu_{A_n}(x_n) \right\}; y = f(x_1,\ldots,x_n) \right] \\ 0 \quad \text{si} \quad f^{-1}(y) = 0 \end{cases}$$

Les nombres flous utilisés dans la régression linéaire floue, sont définis par leur centre, leurs deux supports à gauche et à droite, et par leurs deux fonctions L et R :

$$a_j\left(\Omega_j,\alpha_j,\beta_j\right)_{L\text{-}R}$$

La relation (3) peut aussi se mettre sous la forme suivante (L.Duckstein 1998) :

$$Y_i = a_0 + \sum_{j=1}^{n} a_j \cdot \left(x_{j,i} - \overline{x}_j\right) \qquad (4)$$

Dans cette formulation $\overline{X} = \{\overline{x}_1,\ldots,\overline{x}_n\}$ représente un vecteur de points de référence. Dans la plupart des cas, ces points de référence représentent les moyennes de la série de données de chaque variable explicative.

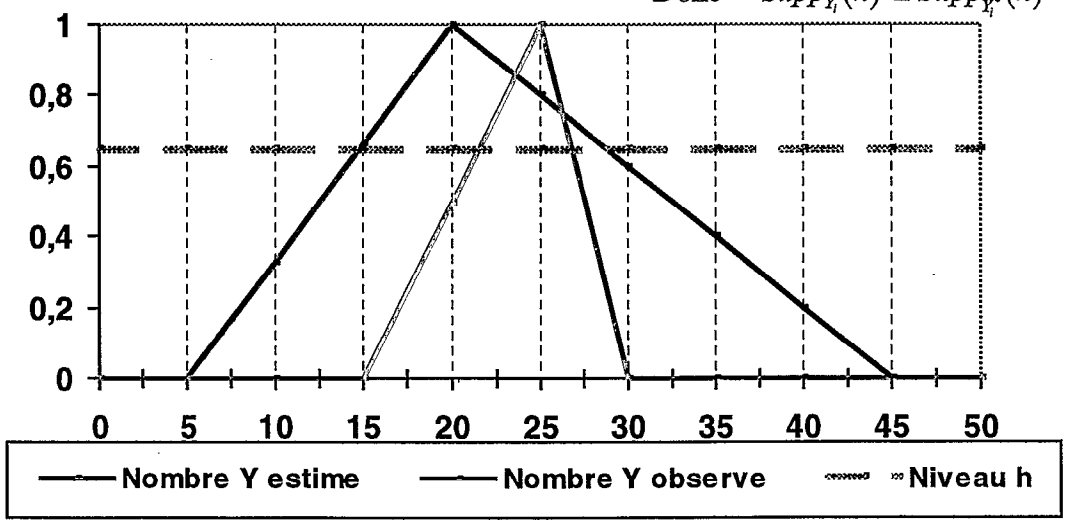
• Le rôle du niveau d'ajustement H .

Il existe en régression floue un niveau d'ajustement du modèle, note 'h' compris entre zéro et un. Il représente le niveau de crédibilité du modèle jugé acceptable si un ajustement du modèle est effectivement réalisé pour ce niveau. La valeur de ce niveau est choisie par l'utilisateur, elle est généralement comprise entre 0,5 et 0,7.

Le niveau 'h' s'exprime comme une contrainte sur les Y_i^* estimés. Le support_h du nombre flou estimé Y_i^* doit contenir le support_h du nombre flou Y_i observé ou la valeur Y_i si la variable expliquée est représentée par des nombres précis.

Voici un exemple qui vérifie cette contrainte : $Supp_{Y_i^*}(h) = [15;28]$ et $Supp_{Y_i}(h) = [21;27]$

Donc $Supp_{Y_i}(h) \subset Supp_{Y_i^*}(h)$



(Figure 2)

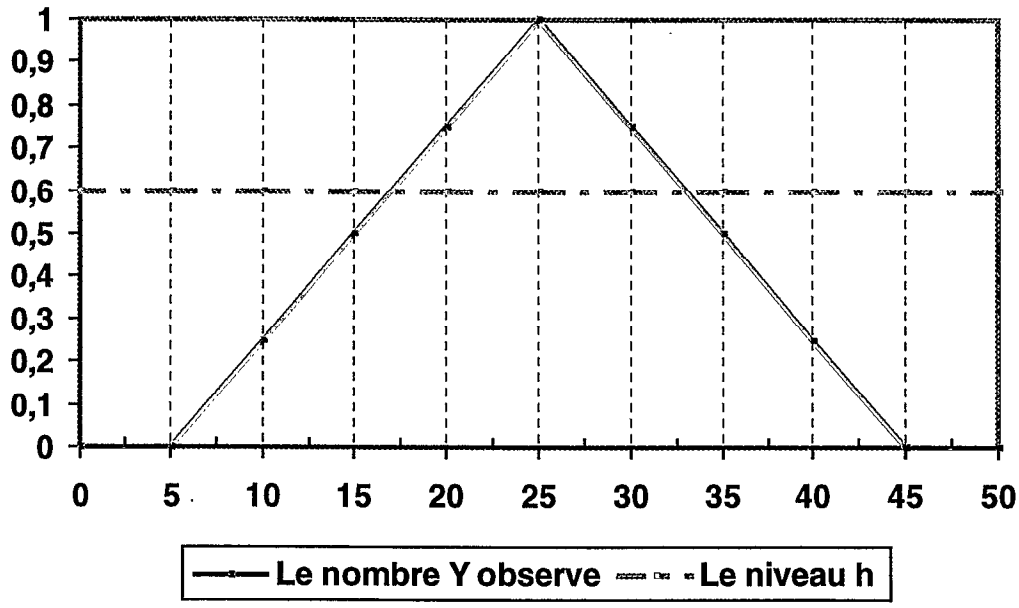
Ces contraintes doivent être vérifiées pour toutes les observations de la variable expliquée par rapport à leurs valeurs estimées. Il existe donc autant de systèmes de contraintes (ce sont des systèmes puisque l'on parle ici de deux inégalités) que d'observations.

• La définition des contraintes.

Par le principe d'extension des nombres flous, s'il existe des paramètres de niveau supérieur ou égal à 'h' vérifiant l'équation de la régression floue pour toutes les observations explicatives, alors chaque estimation de la variable expliquée aura un niveau de crédibilité supérieur ou égal à 'h'. Ainsi les contraintes d'encadrement présentées au paragraphe précédent peuvent s'exprimer en fonction des paramètres A(cf. annexe 1), se traduisant pour chaque observation par un système d'inéquations avec comme inconnu l'ensemble des éléments composant chaque nombre flou de A (leur centre et leurs supports à droite et à gauche).

On note $Min_{Y_i}(h)$ et $Max_{Y_i}(h)$ les valeurs minimum et maximum de l'intervalle défini par Y_i au niveau 'h' (Bardossy et al. 1990).

Dans l'exemple suivant : $Min_{Y_i}(h) = 17$ et $Max_{Y_i}(h) = 33$



(Figure 3)

- ***La régression floue.***

La régression statistique classique a pour but de définir une relation entre des variables appelées explicatives et une variable expliquée. Elle s'appuie sur un modèle théorique supposant des variables aléatoires de nature (ce qui représente la propriété de normalité) et de comportement (ce qui représente les propriétés d'indépendance des variables et d'homoscédasticité , variances constantes) rigoureusement définies. La vérification de ces hypothèses implique de disposer d'une grande séric d'observations. Or dans de nombreux cas pratiques, ces dernières sont insuffisantes en quantité ou en qualité(par des erreurs de mesures) pour recourir à la mise en œuvre de la régression linéaire statistique classique (risque important de distorsions dans l'estimation des paramètres). La régression linéaire floue peut alors être une alternative intéressante. Elle est aussi utilisée lorsque la variable expliquée est représentée en nombres flous.

3. La régression linéaire floue :

- **Son but et sa définition.**

Le but de la régression linéaire floue est de définir une relation entre des séries de variables explicatives et une série de variable expliquée. Les paramètres de cette relation seront des nombres flous, tout comme les résultats estimés de la variable expliquée. Les variables explicatives et la variable expliquée peuvent, suivant les cas être représentées par des nombres flous. Tout au long de notre travail, les variables explicatives seront des nombres précis.
La régression linéaire floue s'écrit sous la forme(A.Bardossy, 1990) :

$$Y_i = a_0 + \sum_{j=1}^{n} a_j \cdot x_{j,i} \qquad (3)$$

Dans cette équation :

Y_i représente la i ème observation de la variable expliquée Y.

$X_i = \{x_{1,i},\ldots,x_{n,i}\}$ représente l'ensemble des variables explicatives $X_1,..,X_n$ pour la i ème observation.

$A = \{a_0,\ldots,a_n\}$ représente l'ensemble des paramètres de la régression.

La répercussion des contraintes définies ci dessus est représentée pour chaque observation de la manière suivante (A.Bardossy 1990) :

$$Min_{Y_i}(h) \geq \Omega_0 - \alpha_0 \cdot L_0^{-1}(h) + \sum_{x_{j,i} \geq \bar{x}_j} \left(\left(\Omega_j - \alpha_j \cdot L_j^{-1}(h) \right) \left(x_{j,i} - \bar{x}_j \right) \right) + \sum_{x_{j,i} < \bar{x}_j} \left(\left(\Omega_j + \beta_j \cdot R_j^{-1}(h) \right) \left(x_{j,i} - \bar{x}_j \right) \right)$$

$$Max_{Y_i}(h) \leq \Omega_0 + \beta_0 \cdot R_0^{-1}(h) + \sum_{x_{j,i} < \bar{x}_j} \left(\left(\Omega_j - \alpha_j \cdot L_j^{-1}(h) \right) \left(x_{j,i} - \bar{x}_j \right) \right) + \sum_{x_{j,i} \geq \bar{x}_j} \left(\left(\Omega_j + \beta_j \cdot R_j^{-1}(h) \right) \left(x_{j,i} - \bar{x}_j \right) \right) \quad (5)$$

Il existe donc dans la plupart des cas une infinité de solutions à ce système d'inéquations.

• *Le degré d'imprécision.*

Le but de la régression floue est de définir une relation linéaire à paramètres flous tout en minimisant l'imprécision des Y_i^* estimés. Ces derniers sont donc recherchés par minimisation d'un critère d'imprécision qui reflète donc la fiabilité du modèle. D'après la litterature, ce critère prend traditionnellement trois formes possibles qui sont les suivantes(Bardossy et al., 1990) :

Le critère d'imprécision maximal :

Il reflète le degré d'imprécision de l'ensemble des nombres flous utilisés dans le modèle. Il fait référence au support maximal de l'ensemble des paramètres. Ce critère est représenté par le support (à gauche ou à droite) le plus grand de tous les paramètres utilisés. En minimisant ce dernier, l'imprécision du nombre flou le plus imprécis est minimisée.

Critère à minimiser : $V = Max\left(\forall j \in [0;n]; Max\left(\alpha_j, \beta_j \right) \right)$ (6)

Le critère d'imprécision par moyenne :

Il fait référence à la moyenne des supports de l'ensemble des paramètres. Il est représenté par la moyenne de tous les supports (à gauche et à droite) de tous les paramètres. En minimisant ce dernier, l'imprécision de tous les paramètres est minimisée en moyenne:

Critère à minimiser : $$V = \left(\frac{1}{2(n+1)}\right) \times \sum_{j=0}^{n} (\alpha_j + \beta_j) \qquad (7)$$

Le critère d'imprécision dit "de prédiction" :

Il fait référence aux aires des fonctions d'appartenance des Y_i estimés (cf. annexe 2).

Critère à minimiser : $$V = \int_{x_n^-}^{x_n^+} \cdots \int_{x_1^-}^{x_1^+} \int_{-\infty}^{+\infty} \left(\max_A \left\{ \min_j \left[\mu_{a_i}(a_i) \right]; y = f(X, A) \right\} \right) dy dx_1 \ldots dx_n \qquad (8)$$

Après quelques simplifications élaborées par A. Bardossy (1990), ce critère s'exprime comme une fonction linéaire de α_i et β_i :

$$\forall i \quad \text{Soit} \quad e_i = x_i^+ - \overline{x}_i \quad ; \quad d_i = \overline{x}_i - x_i^-$$

$$\text{Et} \quad L_i^\circ = \int_0^1 L_i(t)dt \quad ; \quad R_i^\circ = \int_0^1 R_i(t)dt$$

$$V = \left(\alpha_0 \cdot L_0^\circ + \beta_0 \cdot R_0^\circ\right) + \sum_{i=1}^{n} \frac{d_i^{\,2} + e_i^{\,2}}{2(d_i + e_i)} \cdot \left(\alpha_i \cdot L_i^\circ + \beta_i \cdot R_i^\circ\right)$$

- ***Conclusion sur la régression linéaire floue.***

Les estimations de la régression linéaire floue dépendent donc :

- du choix des fonctions L et R qui définissent la forme des nombres flous utilisés,
- du choix du point de référence généralement décrit comme la moyenne de chaque variable explicative,
- du choix du critère à minimiser,
- du choix du niveau 'h' défini comme le degré de calibration choisi par l'utilisateur du modèle,

Les variables expliquées estimées sont donc des nombres flous. Suivant l'utilisation de ces résultats, il est possible d'en déduire des nombres précis d'après l'un des différents algorithmes dits de "déflouification"(cf. annexe 3).

4. Le programme et sa structure.

- ***Les entrées et les sorties.***

Les entrées:

- Le nombre d'observations afin de pouvoir correctement lire le fichier de données.
- Le mode d'utilisation : calibration, validation ou estimation(cf. La régression linéaire statistique).
- Le niveau 'h' de calibration de la régression floue.
- L'ensemble des données des variables explicatives et suivant le mode d'utilisation, la variable expliquée.
- Dans le cas d'une validation ou d'une estimation, les paramètres A_i de la régression sont nécessaires en entrée.

- L'ensemble des p et q des nombres L et R au sens de Dubois et Prade(cf. Les nombres Flous) des paramètres A_i. Ces valeurs sont définies par l'utilisateur lors de la création du modèle de régression linéaire floue.

- Le choix du critère de précision.

- Le choix de l'algorithme de déflouification utilisé.

Les sorties:

- L'ensemble des Y_i estimés flous et une fois déflouifiés.

- Dans le cas d'une calibration, les paramètres A_i.

• Le programme

Présentation globale du programme:

Apres avoir stocké l'ensemble des données, le programme résout le problème de la régression linéaire floue, travaillant sur les supports et les centres des paramètres A_i, en effet ce problème peut se rapporter à une minimisation d'une fonction linéaire sous contraintes. On a donc utilisé un algorithme mettant en œuvre la méthode du Simplex sous contrainte(ref. *Numerical Recipes in Fortran 77)*. Une fois les résultats trouvés, le programme calcule les Y_i estimés avant de les déflouifier et de les comparer aux Y_i observés.

Description plus approfondie du programme:

Une fois les données stockées dans des variables, le programme calcul la moyenne de chaque variable explicative afin d'être utilisée dans les contraintes émises sur les Y_i estimés.

Puis une procédure d'incrémentation du tableau des caractéristiques de la procédure Simplex est appelée. Elle débute par le calcul des caractéristiques de la fonction à minimiser, qui dépend du choix du degré d'imprécision. La fonction la plus complexe et la plus utilisée est celle du degré d'imprécision dit de "prédiction" qui nécessite en plus des données, les moyennes des variables explicatives et les valeurs p et q des fonctions L-R de chaque paramètre A_i. Cette fonction est définie sur les supports des paramètres A_i. Par la suite, le programme prépare le calcul des contraintes émises sur les Y_i estimés. Cet algorithme nécessite les données de la régression ainsi que les moyennes de chaque

variable explicative et les valeurs p et q des fonctions Let R de chaque paramètre A_i. Ces contraintes sont définies par rapport aux supports et aux centres des paramètres A_i de la régression linéaire floue.

Ensuite, le programme appelle la procédure Simplex.

Les paramètres A_i étant établis sont sauvegardés dans un fichier afin d'être réutilisés lors de la validation du modèle , puis lors de l'estimation de nouveaux Y_i. Le programme calcule ensuite les Y_i estimés en tant que nombres flous et les représente, ainsi que les Y_i observés, à l'aide d'un outil graphique disponible au laboratoire d'hydrologie, programmé en Tcl/Tk. Une fois les résultats émis graphiquement, le programme déflouifie les nombres estimés avant de les stocker dans un fichier pour une utilisation statistique ultérieure.

• La structure du programme.

Le programme principal de la régression:

Il appelle successivement les procédures suivantes:

- LireFic,
- VarTriangle
- CalculMoyenne
- RemplirTab
- Simplex
- ResultSimplex
- Estimation
- Graphique
- Résultat

La procédure LireFic:

Elle lit le fichier paramètre contenant toutes les entrées de la régression avant d'appeler suivant les valeurs contenues dans le fichier d'autres procédures de lecture de fichiers. Ces dernières prennent en compte les données de la régression, les valeurs des p et q des fonctions L-R des paramètres A_i, ou suivant le mode utilisé les valeurs des ces paramètres.

La procédure VarTriangle:

Elle remplit le tableau des p et q des fonctions L-R des paramètres A_i lorsque le booléen triangle est vérifié et que le mode utilisé est la calibration. En effet si l'utilisateur décide de travailler avec des paramètres représentés par des nombres flous triangulaires, toutes les valeurs p et q seront égales à un.

La procédure CalculMoyenne

Cette procédure calcule les valeurs des points de références utilisés lors du calcul des différentes contraintes. Ce point est représenté par la moyenne de chaque variable explicative X_i.

La procédure RemplirTab:

Elle se décompose en trois appels de procédure afin de remplir le tableau qui sera utilisé dans la procédure du Simplex:

- FonctionMinimiser qui calcule les différentes valeurs des composants de la fonction à minimiser. La fonction choisie est le critère d'imprécision dit "de prédiction" , en effet ce dernier obtient des résultats plus probants puisque qu'il prend en compte plus de paramètres que les autres critères définis(cf. Le Critère d'imprécision).

- ContInf qui calcule les valeurs des caractéristiques des contraintes posées sur le Min $_{Yi}(h)$ (cf. Définition des Contraintes) à partir des observations, des fonctions L-R des paramètres A_i, et de la moyenne des variables explicatives X_i.

- ContSup qui fait de même que ContInf mais pour les contraintes posées sur le Max $_{Yi}(h)$ (cf. Définition des Contraintes).

La procédure Simplex:

Cette procédure met en œuvre l'algorithme du Simplex permettant de minimiser une fonction linéaire sous contraintes d'infériorités, supériorités et égalitaires. Elle provient du "*Numerical Recipes of Fortran 77*".

La procédure ResultSimplex:

Elle remplit le tableau des paramètres A_i suivant les solutions, lorsqu'elles existent, de la procédure Simplex. Puis cette procédure affecte au tableau A_i les valeurs p et q de leur fonction L-R.

La procédure Estimation:

Cette procédure calcule les estimations de la régression linéaire floue pour chaque observation en utilisant "Modflou", un programme pré existant du laboratoire d'hydrologie, utilisant les nombres flous. Elle fait appel aux procédures suivantes:

- Fnumb2xy qui discrètise un nombre flou en un ensemble de points afin de pouvoir y effectuer des opérations.

- Fsetop qui établit une opération entre deux nombres flous discrétisés, dans notre cas, nous utiliserons principalement l'addition de nombre flou.

- Deflou qui permet de déflouifier un nombre flou discrétisé par une des différentes méthodes de déflouification(cf. Annexe3).

- La procédure Graphique: Cette procédure affiche à l'écran les nombres flous Y_i estimés pour chaque observation ainsi que les Y_i observés lorsque le mode d'utilisation est la calibration ou la validation d'un modèle. Pour ce faire, nous utilisons un outil graphique de " Modflou " écrit en Tcl/Tk permettant de visualiser la fonction d'appartenance d'un nombre flou.

La procédure Résultat

Cette dernière procédure écrit dans les fichiers résultats passés en paramètres les Y_i estimés ainsi que les paramètres A_i lorsque l'on se trouve en calibration.

• La structure des fichiers.

Le programme est composé de cinq fichiers comportant toutes les procédures nécessaires, sauf évidement des procédures utilisées se référant à " Modflou ".

- Régression.f : ce fichier contient le programme principal.

- LireFic.f : qui contient l'ensemble des procédures de lecture de fichier.

- Procédure.f : ce fichier est composé de toutes les procédures de calcul antérieures à l'utilisation de la méthode du Simplex

- Simplex.f: qui contient donc une procédure de la méthode du Simplex.

- Résultat.f qui comporte toutes les procédures de calcul de résultats ultérieures à l'utilisation de la méthode du Simplex.

- **La conclusion du programme.**

A ce jour le programme de régression linéaire flou est complet mais contient quelques erreurs de telle sorte que les Y_i estimés ne sont pas similaires aux résultats des exemples choisis. De ce fait, et par la limite de la durée du stage les tests de comparaison avec la régression linéaire statistique n'ont pu aboutir. Malgré tout, la structure du programme et ses nombreux commentaires permettront, je l'espère, de combler ce manque et de développer les autres aspects de la régression linéaire floue, très intéressants mais qui auraient requis un stage d'une durée plus importante.

Conclusion

Durant ce stage de quatre mois, notre travail s'est partagé en deux grandes parties, la première a porté sur la découverte et la compréhension du flou et plus profondément sur la régression linéaire floue. La seconde a débuté par un apprentissage du langage Fortran suivi de la programmation d'un logiciel de calcul de la régression floue.

Ce stage m'a permis de découvrir certains aspect du monde du travail, de la recherche et de l'informatique. Effectivement, l'important travail de recherche effectue m'a permis d'améliorer et de structurer mes méthodes de travail alors que l'utilisation du Fortran m'a obligé à réfléchir différemment sur l'approche d'un problème informatique par rapport aux langages étudiés à l'I.U.P. et m'a fait découvrir mon manque d'organisation lors de la mise en place de mon travail.

Pour moi ce stage a été l'occasion d'être confronté à de nombreux problèmes que l'on ne rencontre pas lors des études, avec une certaine responsabilité. Il m'a permis aussi de découvrir un nouvelle discipline très intéressante : le flou, dont je ne soupçonnais pas l'existence, mais qui rythme malgré tout notre quotidien.

Bibliographie :

- Bardossy A. 1990. *Note on fuzzy regression,* Elsevier Science Publishers B.V. 0165-0114.

- Bardossy A., Duckstein L., Bogardi I. 1990. *Fuzzy regression in hydrology,* Water resources research Vol. 26 No7 Pages 1497-1508.

- Bardossy A., Hagaman R., Duckstein L., Bogardi I. 1992. *Fuzzy least square regression : Theory and application,* Fuzzy regression analysis p181-193.

Dagnelie P. 1988. *Théorie et méthodes statistiques,* Presse agronomique de Gembloux, vol.1 et vol.2.

- Duckstein L. 1998. *Use of fuzzy logic to encode archival climate research uncertainty,* Technica Document in Hydrologie No 17.

- Kim K.J., Moskowitz H., Koksalan M. 1996. *Fuzzy versus statistical linear regression,* Elsevier Science B.V. 0377-2217.

- Pesti G., Boreux J-J., Duckstein L., Nicolas J. 1994. *Fuzzy regression in climatological studies : radiocarbon dating by layer depth,* JGRD 2468.

- Peters G. 1994. *Fuzzy linear regression with fuzzy intervalls,* Elsevier Science B.V. 0165-0114.

- Teukolsky S.A., Vetterling W.T., Flannery B.P. 1992 *Numerical Recipes in Fortran 77,* The art of the computing, 2nd edition.

- Tanaka H., Uejima S., Asai K. 1982. *Linear regression analysis with Fuzzy Model,* IEEE Transaction on systems, man, and cybernetics Vol. SMC-12 No. 6.

- Tong-Tong, J-R. 1995. *La logique floue,* Hermes, Paris.

Annexe 1 : Les contraintes de la régression linéaire floue.

D'après les contraintes imposées au support_h des valeurs de la variable observée Y_i par rapport aux valeurs de la variable estimée Y_i^* :

$$Supp_{Y_i}(h) \subset Supp_{Y_i^*}(h)$$

Donc $\quad Min_{Y_i}(h) \geq Min_{Y_i^*}(h) \quad$ et $\quad \mathrm{Max}_{Y_i}(h) \leq Max_{Y_i^*}(h)$

De plus pour tout nombre flou A :

si $\quad L_A\left(\dfrac{\Omega_A - Min_A(h)}{\alpha_A}\right) = h$

alors $\quad L_A^{-1}(h) = \dfrac{\Omega_A - Min_A(h)}{\alpha_A}$

Donc: $\quad Min_A(h) = \Omega_A - \alpha_A \cdot L_A^{-1}(h)$

De même pour le maximum : $\quad Max_A(h) = \Omega_A + \beta_A \cdot R_A^{-1}(h)$

Donc les contraintes peuvent se formuler de la manière suivante :

$$Min_{Y_i}(h) \geq \Omega_{Y_i^*} - \alpha_{Y_i^*} \cdot L_{Y_i^*}^{-1}(h) \quad \text{et} \quad Max_{Y_i}(h) \leq \Omega_{Y_i^*} + \beta_{Y_i^*} \cdot R_{Y_i^*}^{-1}(h)$$

Ses contraintes se répercutent donc sur les paramètres A, par rapport à l'équation de la régression linéaire :

$$Min_{Y_i}(h) \geq \Omega_0 - \alpha_0 \cdot L_0^{-1}(h) + \sum_{x_{j,i} \geq \bar{x}_j} \left(\left(\Omega_j - \alpha_j \cdot L_j^{-1}(h) \right) \left(x_{j,i} - \bar{x}_j \right) \right) + \sum_{x_{j,i} < \bar{x}_j} \left(\left(\Omega_j + \beta_j \cdot R_j^{-1}(h) \right) \left(x_{j,i} - \bar{x}_j \right) \right)$$

$$Max_{Y_i}(h) \leq \Omega_0 + \beta_0 \cdot R_0^{-1}(h) + \sum_{x_{j,i} < \bar{x}_j} \left(\left(\Omega_j - \alpha_j \cdot L_j^{-1}(h) \right) \left(x_{j,i} - \bar{x}_j \right) \right) + \sum_{x_{j,i} \geq \bar{x}_j} \left(\left(\Omega_j + \beta_j \cdot R_j^{-1}(h) \right) \left(x_{j,i} - \bar{x}_j \right) \right)$$

Annexe 2 : Le critère d'imprécision dit "de prédiction" de la régression linéaire floue.

Le critère d'imprécision dit "de prédiction" est une mesure globale de l'imprécision de la régression linéaire floue. L'imprécision d'un nombre flou est décrite par la grandeur de la surface de sa fonction d'appartenances. Ce critère fait donc référence aux surfaces décrites par les fonctions d'appartenances des Yi estimés.

Pour un vecteur X_i donné (X_i est le vecteur des valeurs des variables explicatives de l'observation i), une mesure d'imprécision $H(X_i)$ peut être calculée comme l'aire de la fonction d'appartenance du nombre flou estimé a partir de la régression.

$$H(x_{1,i},\ldots,x_{n,i}) = \int_{-\infty}^{+\infty} \mu(x_{1,i},\ldots,x_{n,i},y_i)dy$$

$$\mu(x_{1,i},\ldots,x_{n,i},y_i) = \max_A\left\{\min_j\left[\mu_{a_i}(a_i)\right]; y = f(X,A)\right\} \qquad \text{(Par le principe d'extension)}$$

Afin de traiter tout le domaine de définition de chaque variable explicative, on définit :

$$\forall j \in [1,N] \quad \exists\ x_j^- \text{ et } x_j^+ \text{ tel que } \forall i \quad x_j^- \le x_{j,i} \le x_j^+$$

De ce fait le critère d'imprécision global est représenté par l'hypervolume défini par l'ensemble des fonctions d'appartenance des Y_i estimés. Il se formule de la manière suivante :

$$V = \int_{x_1^-}^{x_1^+} \ldots \int_{x_n^-}^{x_n^+} H(X)\cdot dx_n \ldots dx_1$$

Après quelques simplifications élaborées par A. Bardossy (1990), ce critère s'exprime comme une fonction linéaire :

$$\forall i \quad \text{Soit} \quad e_i = x_i^+ - \overline{x}_i \quad ; \quad d_i = \overline{x}_i - x_i^-$$

$$\text{Et} \quad L_i^\circ = \int_0^1 L_i(t)dt \quad ; \quad R_i^\circ = \int_0^1 R_i(t)dt$$

$$V = \left(\alpha_0 \cdot L_0^\circ + \beta_0 \cdot R_0^\circ\right) + \sum_{i=1}^{n} \frac{d_i^{\,2} + e_i^{\,2}}{2(d_i + e_i)} \cdot \left(\alpha_i \cdot L_i^\circ + \beta_i \cdot R_i^\circ\right)$$

Annexe 3 : Les différents algorithmes dit de "déflouification".

Il existe plusieurs méthodes de transformation des nombres flous en nombres précis, afin de pouvoir utiliser les résultats de la régression linéaire floue en terme de statistique classique.

Trois algorithmes ont été élaborés :

1. Le centre de gravité : Un nombre flou A a pour valeur précise 'p' la projection sur l'axe horizontal du centre de gravité de sa fonction d'appartenance.

$$p = \frac{\int_u u \cdot \mu_A(u) du}{\int_u \mu_A(u) du}$$

2. La moyenne des maxima : La valeur précise d'un nombre flou A est la moyenne de ces éléments dont la fonction d'appartenance est égale à 1.

$$p = \frac{\sum_u (u; \mu_A(u) = 1)}{\text{Nbr de } u \ / \ \mu_A(u) = 1}$$

3. Le point médian : La fonction d'appartenance de tout nombre flou A est séparée en deux surfaces égales par la droite d'équation y=p ou p est le point médian dc A.

$$\int_{-\infty}^{p} \mu_A(u) du = \int_{p}^{+\infty} \mu_A(u) du$$

Annexe 4:

Le code du programme

```
      program regression

      include 'info.ins'

c     declaration
c     -----------



      integer nobs,nobs2,yflou,triangle,critere,nvar,npar1,npar2,deflou
      integer icase,izrov(LongTab),iposv(LargTab)

      real obs(nvar5,nobs0),pq(2,nvar1),Ai(5,nvar1)
      real h,p,q,moyenne(nvar0)
      real tab(LargTab,LongTab),estim(nobs0)

      character*128 nomfic,ficdonnees,interm,ficflou,ficresult,ficpq
      character*3 mode


c     main
c     -----
      character*128 nomfic
      nomfic='fichier.par'
      call lirefic(nomfic,nobs,nvar,mode,Ai,h,yflou,obs,ficdonnees,
     &     triangle,ficpq,pq,p,q,critere,deflou,interm,
     &     ficflou,ficresult)

      if (triangle.eq.1) then
         if (mode.eq.'cal') then
            call VarTriangle(nvar,pq)
         endif
      end if
      print *, 'nombres d observationn',nobs
      print *, 'nombres de variables..', nvar
      print *, 'Le mode ',mode
      print*, 'Les Ai'
      print '(5f)', Ai
      print *, 'La hauteur' , h
      print *, 'y est flou', yflou
      print *, 'fichier de donnees', ficdonnees
      print *, ' Les obs'
      print '(8f)', obs
      print *, 'triangulaire', triangle
      print '(2f)', pq
      print *, 'P',p
      print *, 'Q',q
      print *, 'Critere' , critere
      print *, 'deflouification', deflou
      print *, 'interm', interm
      print *, 'fichier resultat',ficresult


      if (mode.eq.'cal') then
         call calculmoyenne(obs,nvar,nobs,moyenne)
         print *,' Les moyennes ',moyenne
```

```
      call RemplirTab(tab,obs,nvar,nobs,h,pq,yflou,moyenne)
      print *, ' Simplex !!'
      npar1=2*nobs
      npar2=3*(nvar+1)
      nobs2=nobs

      print *, ' par1 ' ,npar1, ' npar2 ',   npar2
      print *,'LargTab ',LargTab,' LongTab ',LongTab,' nobs ',nobs
      call simplx(tab,npar1,npar2,LargTab,LongTab,nobs,
     &     nobs2,0,icase,izrov,iposv)


      print *, 'NEW TAB:  '
      print '(32f19.3)', tab
      print *, ' IZROV:  '
      print *, izrov
      print *, ' IpOsV:  '
      print *, iposv
      print *, ' Icase:  '
      print *, icase

      print *,' test'
      call resultsimplex(tab,icase,izrov,iposv,Ai,npar1,
     &     npar2,pq,nvar)
      print '(5f)', Ai

      endif
      call estimation(nobs,nvar,obs,Ai,deflou,estim,yflou)
      call Resultat(Ai,estim,nobs,nvar,Mode,ficflou,ficresult)

      end
```

```
# entrees
# --------
# nombre d'observations
14
# Mode d'utilisation (Cal,Val,Est)
cal
# Hauteur de la regresion
0.75
# Les Yi observes sont flou ->0  sinon 1
0
# Nom du fichier contenant les donnees de la regression
donnee.par
# Nom du fichier contenant les parametres ( sauf si on est en calibration)
# Si on est en calibration: Les parametres Ai sont triangulaire ->1 sinon 0
1
# Nom du ficheir contenant les p et q des ai ( sauf si ils sont triangulaires)
#pq.par
# le p des Yi a estimer
1
# le q des Yi a estimer
1
# algo
# -----
# choix du critere d'imprecision (max=1 , moy=2, pred=3)
3
# choix de l'algorithme de deflouification(grav=1 , max=2 , med=3)
1
# resultats
#---------
# nom du fichier intermediaire
interm.par
# nom du fichier contenant les Yi estimes
result.par
# nom du fichier contenant les Ai de la regression( seulement en calibration)
ficflou.par
```

```
      subroutine lirefic(nomfic,nobs,nvar,mode,Ai,h,yflou,obs,ficdonnees
     &     ,triangle,ficpq,pq,p,q,critere,deflou,interm,ficflou,
     &     ficresult)

      include 'info.ins'
c Les entrees:
c -----------
c     nomfic    : le nom du fichier a lire
c     nobs      : le nombre d'observations
c     mode      : Le mode de traitement (cal=calibration, val=validation, est=estimat
ion)
c     Ai        : Le tableau des parametres flous Ai de la regression ( il peut se re
trouver en sortie si le  mode est en calibration)
c     h         : La hauteur de la regression (son niveau)
c     yflou     : Le boolean verifiant si les y observes sont des nombres flous
c     obs       : Le tableau de donnees (d'observations)
c     ficdonnees: Le nom du fichier de donnees de la regression
c     triangle  : Le boolean verifiant si les ai sont des nombres triangulaires
c     ficpq     : Si il ne sont pas triangulaire leur definition est dans ce fichier
c     p,q       : les deux nombrse definisant les nombres flous yi observes
c
c Les Algos:
c ---------
c     critere   : Le critere choisi (1,2 ou 3 et 0 pour les trois en meme temps)
c     deflou    : L'alogorithme de deflouification choisi (max, cdg ou med)
c
c Les resultats:
c -------------
c     nvar      : Le nombre de variables explicatives
c     interm    : Le nom d'un fichier intermediaire
c     ficresult: Le nom du fichier resultat
c     ficflou   : Le nom du fichier des ai (uniquement en mode calibration)


c     declaration
c     -----------
      integer  nobs,yflou,triangle,critere,nvar,iul,deflou

      real obs(nvar5,nobs0),pq(2,nvar1),h,p,q,Ai(5,nvar0)
      character*128 nomfic,ficpq,ficdonnees,interm,ficflou,ficresult,str
      character mode*3,er*128

c     entrees
c     -------

      call ouvrir(nomfic,iul)


      call suivant(iul,str,er)
      read(str,*) nobs

      call suivant(iul,str,er)
      read(str,*) mode


      call suivant(iul,str,er)
      read(str,*) h
```

```
      call suivant(iul,str,er)
      read(str,*) yflou


      call suivant(iul,ficdonnees,er)
      call LireDonnees(ficdonnees,nobs,obs,nvar,mode,yflou)


      if (mode.ne.'cal') then
         call suivant(iul,ficflou,er)
         call LireAi(ficflou,nvar,Ai)
      end if

      call suivant(iul,str,er)
      read(str,*) triangle
      if (triangle.ne.1) then
         call suivant(iul,ficpq,er)

         call lirepq(ficpq,nvar,pq)
      endif

      call suivant(iul,str,er)
      read(str,*) p

      call suivant(iul,str,er)
      read(str,*) q


c     Algos
c     -----

      call suivant(iul,str,er)
      read(str,*) critere

      call suivant(iul,str,er)
      read(str,*) deflou

c     Resultats
c     ---------

      call suivant(iul,interm,er)

      call suivant(iul,ficresult,er)

      if (mode.eq.'cal') then
         call suivant(iul,ficflou,er)
      endif

      end


c -------------------------------------

      subroutine suivant(iul,str,er)

c     Teste retourne la chaine de caractere suivante ne commencant pas par un #
c
c     Entree
```

```
c     -----
c     iul    : l'ul sur le fichier a lire
c
c     Sortie
c     ------
c     str    : la chaine de caractere retournee
c     er     : Un code d'erreur
c ----------------------------------------
      integer iul
      character *128 str,er
      er= 'OK'
      read(iul, '(a)', err=99, end=9) str

c     On traite la chaine pour retirer les caracteres blancs autour
      call string(str, lstr, ils0, ils1)
      str = str(ils0:ils1)

      do while (str(1:1).eq.'#')
         read(iul, '(a)', err=99, end=9) str
         print *, ' STR ' ,str
         call string(str, lstr, ils0, ils1)
         str = str(ils0:ils1)
      end do
      return
c     Gestion des erreurs
 9    write(*,*) 'suivant:fin de fichier', str
      er='eof'
      return
 99   write(*,*) 'suivant:erreur'
      stop
      end



c ---------------------------------------
      subroutine  LireDonnees(ficdonnees,nobs,obs,nvar,mode,yflou)

c     Lecture du fichier comportant toutes les donnes : le valeurs des variables expl
icatives et suivant le mode celle de la variable explique ( cette derniere peut etre
floue)l
c     Entree
c     ------
c     ficdonnees : Le nom du fichier contenant ces donnees
c     nobs       : Le nombre d'observations
c     yflou      : un entier =1 si la variable y est flou sinon =0
c     mode       : Le mode de regression: calibration(cal),validation(val) ou estimat
ion(est)
c
c     Sortie
c     ------
c     obs      : Un tableau de reels contennanttoutes cesdonnees
c     nvar     : Un entier representant le nombre de variables explicatives
c     ---------------------------------------


      include 'info.ins'
      integer nobs,nvar,yflou,k,i,iul
      real obs(nvar5,nobs0)
      character ficdonnees*128,mode*3,str*128,er*128
      real val(nvar0+5)
```

```
      integer IST2

      k=1
      call ouvrir(ficdonnees,iul)

      do while  (k.ne.nobs+1)
         call suivant(iul,str,er)
c     traitement de la chaine de caracteres contenu dans str
         call STRINGREAL(str,nvar,val,IST2)
         i=1
c     affectation des valeur des val2 dans le tableau obs
         do while (i.ne.nvar+1)
            obs(i,k)=val(i)
            i=i+1
         end do

         k=k+1
      enddo

c     Cacul de nvar
c     -------------

      if (mode.ne.'est') then
         if (yflou.ne.0) then
            nvar=nvar-5
         else
            nvar=nvar-1
         end if
      end if
c     -------------
      end




c ----------------------------------------
      subroutine lirepq(ficpq,nvar,pq)

c     Lecture du fichier comportant les valeurs des fonctions LR des parametres Ai
c     Entree
c     ------
c     ficfpq  : Le nom du fichier contenant ces valeurs
c     nvar    : Le nombre de variable
c
c     Sortie
c     ------
c     pq      : Un tableau de reels contennant les valeurs des fonctions LR des  para
metres Ai
c     ----------------------------------------

      include 'info.ins'
      integer  nvar,i,j,iul
      real  pq(2,nvar0)
      character*128 ficpq,str,er

      i=1
```

```
      call ouvrir(ficpq,iul)
      do while (i.ne.nvar+1)
         call suivant(iul,str,er)
         read(str,*) (pq(j,i), j=1, 2)
         i=i+1
      end do
      end




c -----------------------------------------

      subroutine lireAi(ficflou,nvar,Ai)


c     Lecture du fichier comportant les parametres flous
c     Entree
c     ------
c     ficflou : Le nom du fichier contenant les parametres
c     nvar    : Le nombre de parametres
c
c     Sortie
c     ------
c     Ai      : Un tableau d'entier contennant toutes les informations sur les parame
tres Ai
c     ---------------------------------------


      include 'info.ins'
      integer nvar,i,j,iul
      real Ai(5,nvar0)
      character*128 ficflou,str,er
      i=1

      call ouvrir(ficflou,iul)
c     Lecture de tout le fichier
      do while (i.ne.nvar+1)
         call suivant(iul,str,er)
         read(str,*) (Ai(j,i), j=1, 3)
         i=i+1
      end do

      end




c -----------------------------------------
        subroutine ouvrir(name, iul)
```

```
c
c     Teste et ouvre le fichier nom
c
c     Entree
c     -----
c     name : Le nom du fichier a ouvrir
c
c     Sortie
c     ------
c     iul  : une ul libre
c ----------------------------------------
        character*128 name
        integer*4 iul

c Test du fichier
        call testfil('old', name, iret)
        if (iret.ne.0) then
c fichier inexistant
           stop
        endif
c ON cherche une unite disponible pour le fichier
        call ul(iul)
c Ouverture du fichier
        open(unit=iul, file=name, err=99, status='old')
        print *, name(1:15), 'ok!'
        return

99      print *, name(1:15), ' : Erreur ouverture '
        stop
        end
```

```
      subroutine lirefic(nomfic,nobs,nvar,mode,Ai,h,yflou,obs,ficdonnees
     &     ,triangle,ficpq,pq,p,q,critere,deflou,interm,ficflou,
     &     ficresult)

      include 'info.ins'
c Les entrees:
c -----------
c     nomfic    : le nom du fichier a lire
c     nobs      : le nombre d'observations
c     mode      : Le mode de traitement (cal=calibration, val=validation, est=estimat
ion)
c     Ai        : Le tableau des parametres flous Ai de la regression ( il peut se re
trouver en sortie si le  mode est en calibration)
c     h         : La hauteur de la regression (son niveau)
c     yflou     : Le boolean verifiant si les y observes sont des nombres flous
c     obs       : Le tableau de donnees (d'observations)
c     ficdonnees: Le nom du fichier de donnees de la regression
c     triangle  : Le boolean verifiant si les ai sont des nombres triangulaires
c     ficpq     : Si il ne sont pas triangulaire leur definition est dans ce fichier
c     p,q       : les deux nombrse definisant les nombres flous yi observes
c
c Les Algos:
c ---------
c     critere   : Le critere choisi (1,2 ou 3 et 0 pour les trois en meme temps)
c     deflou    : L'alogorithme de deflouification choisi (max, cdg ou med)
c
c Les resultats:
c -------------
c     nvar     : Le nombre de variables explicatives
c     interm   : Le nom d'un fichier intermediaire
c     ficresult: Le nom du fichier resultat
c     ficflou  : Le nom du fichier des ai (uniquement en mode calibration)


c     declaration
c     -----------
      integer  nobs,yflou,triangle,critere,nvar,iul,deflou

      real obs(nvar5,nobs0),pq(2,nvar1),h,p,q,Ai(5,nvar0)
      character*128 nomfic,ficpq,ficdonnees,interm,ficflou,ficresult,str
      character mode*3,er*128

c     entrees
c     -------

      call ouvrir(nomfic,iul)


      call suivant(iul,str,er)
      read(str,*) nobs

      call suivant(iul,str,er)
      read(str,*) mode


      call suivant(iul,str,er)
      read(str,*) h
```

```
      call suivant(iul,str,er)
      read(str,*) yflou


      call suivant(iul,ficdonnees,er)
      call LireDonnees(ficdonnees,nobs,obs,nvar,mode,yflou)


      if (mode.ne.'cal') then
         call suivant(iul,ficflou,er)
         call LireAi(ficflou,nvar,Ai)
      end if

      call suivant(iul,str,er)
      read(str,*) triangle
      if (triangle.ne.1) then
         call suivant(iul,ficpq,er)

         call lirepq(ficpq,nvar,pq)
      endif

      call suivant(iul,str,er)
      read(str,*) p

      call suivant(iul,str,er)
      read(str,*) q


c     Algos
c     -----

      call suivant(iul,str,er)
      read(str,*) critere

      call suivant(iul,str,er)
      read(str,*) deflou

c     Resultats
c     ---------

      call suivant(iul,interm,er)

      call suivant(iul,ficresult,er)

      if (mode.eq.'cal') then
         call suivant(iul,ficflou,er)
      endif

      end


c ---------------------------------------

      subroutine suivant(iul,str,er)

c     Teste retourne la chaine de caractere suivante ne commencant pas par un #
c
c     Entree
```

```
c     -----
c     iul    : l'ul sur le fichier a lire
c
c     Sortie
c     ------
c     str  : la chaine de caractere retournee
c     er   : Un code d'erreur
c ----------------------------------------
      integer iul
      character *128 str,er
      er= 'OK'
      read(iul, '(a)', err=99, end=9) str

c     On traite la chaine pour retirer les caracteres blancs autour
      call string(str, lstr, ils0, ils1)
      str = str(ils0:ils1)

      do while (str(1:1).eq.'#')
         read(iul, '(a)', err=99, end=9) str
         print *, ' STR ' ,str
         call string(str, lstr, ils0, ils1)
         str = str(ils0:ils1)
      end do
      return
c     Gestion des erreurs
 9    write(*,*) 'suivant:fin de fichier', str
      er='eof'
      return
 99   write(*,*) 'suivant:erreur'
      stop
      end



c ---------------------------------------
      subroutine  LireDonnees(ficdonnees,nobs,obs,nvar,mode,yflou)

c     Lecture du fichier comportant toutes les donnes : le valeurs des variables expl
icatives et suivant le mode celle de la variable explique ( cette derniere peut etre
floue)l
c     Entree
c     ------
c     ficdonnees : Le nom du fichier contenant ces donnees
c     nobs       : Le nombre d'observations
c     yflou      : un entier =1 si la variable y est flou sinon =0
c     mode       : Le mode de regression: calibration(cal),validation(val) ou estimat
ion(est)
c
c     Sortie
c     ------
c     obs      : Un tableau de reels contennanttoutes cesdonnees
c     nvar     : Un entier representant le nombre de variables explicatives
c     ---------------------------------------


      include 'info.ins'
      integer nobs,nvar,yflou,k,i,iul
      real obs(nvar5,nobs0)
      character ficdonnees*128,mode*3,str*128,er*128
      real val(nvar0+5)
```

```
      integer IST2

      k=1
      call ouvrir(ficdonnees,iul)

      do while  (k.ne.nobs+1)
         call suivant(iul,str,er)
c     traitement de la chaine de caracteres contenu dans str
         call STRINGREAL(str,nvar,val,IST2)
         i=1
c     affectation des valeur des val2 dans le tableau obs
         do while (i.ne.nvar+1)
            obs(i,k)=val(i)
            i=i+1
         end do

         k=k+1
      enddo

c     Cacul de nvar
c     -------------

      if (mode.ne.'est') then
         if (yflou.ne.0) then
            nvar=nvar-5
         else
            nvar=nvar-1
         end if
      end if
c     -------------
      end




c ----------------------------------------
      subroutine lirepq(ficpq,nvar,pq)

c     Lecture du fichier comportant les valeurs des fonctions LR des parametres Ai
c     Entree
c     ------
c     ficfpq  : Le nom du fichier contenant ces valeurs
c     nvar    : Le nombre de variable
c
c     Sortie
c     ------
c     pq      : Un tableau de reels contennant les valeurs des fonctions LR des  para
metres Ai
c     ----------------------------------------

      include 'info.ins'
      integer  nvar,i,j,iul
      real  pq(2,nvar0)
      character*128 ficpq,str,er

      i=1
```

```
      call ouvrir(ficpq,iul)
      do while (i.ne.nvar+1)
         call suivant(iul,str,er)
         read(str,*) (pq(j,i), j=1, 2)
         i=i+1
      end do
      end




c -----------------------------------------

      subroutine lireAi(ficflou,nvar,Ai)


c     Lecture du fichier comportant les parametres flous
c     Entree
c     ------
c     ficflou : Le nom du fichier contenant les parametres
c     nvar    : Le nombre de parametres
c
c     Sortie
c     ------
c     Ai      : Un tableau d'entier contennant toutes les informations sur les parame
tres Ai
c     ---------------------------------------


      include 'info.ins'
      integer nvar,i,j,iul
      real Ai(5,nvar0)
      character*128 ficflou,str,er
      i=1

      call ouvrir(ficflou,iul)
c     Lecture de tout le fichier
      do while (i.ne.nvar+1)
         call suivant(iul,str,er)
         read(str,*) (Ai(j,i), j=1, 3)
         i=i+1
      end do

      end




c -----------------------------------------
        subroutine ouvrir(name, iul)
```

```
c
c     Teste et ouvre le fichier nom
c
c     Entree
c     -----
c     name : Le nom du fichier a ouvrir
c
c     Sortie
c     ------
c     iul  : une ul libre
c ----------------------------------------
        character*128 name
        integer*4 iul

c Test du fichier
        call testfil('old', name, iret)
        if (iret.ne.0) then
c fichier inexistant
           stop
        endif
c ON cherche une unite disponible pour le fichier
        call ul(iul)
c Ouverture du fichier
        open(unit=iul, file=name, err=99, status='old')
        print *, name(1:15), 'ok!'
        return

99      print *, name(1:15), ' : Erreur ouverture '
        stop
        end
```

```
c     ----------------------------------------
      subroutine calculmoyenne(obs,nvar,nobs,moyenne)

c     Calcul des moyennes de chaque variable Xi
c
c     Entree
c     ------
c     obs      : Le tableau contenant l'ensemble des observations
c     nvar     : Le nombre de variable
c     nobs     : Le nombre d'observations
c
c     Sortie
c     ------
c     moyenne : Un tableau a une dimensions contenant toutes les moyennes des variabl
es Xi
c     ----------------------------------------
      include 'info.ins'

      integer nvar,nobs,i,j
      real obs(nvar5,nobs0),moyenne(nvar0),moy
c     real min, max
      i=1
      do while (i.ne.nvar+1)
         j=1
         moy=0
         do while (j.ne.nobs+1)
            moy=moy+obs(i,j)
c            print *,'i: ',i, '   j: ',j ,'   obs(i,j):  ',obs(i,j)
            j=j+1
         end do
         moyenne(i)=moy/nobs
         i=i+1
      end do

c     ----------------------------------------------------------------------'
c     Un exemple ou le point de reference est la moyenne entre le point
c     le plus eleve et celui le plus bas de chaque variables explicative.
c     ----------------------------------------------------------------------'
c      do while (i.ne.nvar+1)
c         j=1
c         max=obs(i,j)
c         min=obs(i,j)
c         do while(j.ne.nobs+1)
c            if (obs(i,j).gt.max) then
c               max=obs(i,j)
c            else
c               if (obs(i,j).lt.min) then
c                  min=obs(i,j)
c               endif
c            endif
c            j=j+1
c         enddo
c         moyenne(i)=(max+min)/2
c         i=i+1
c      enddo


      end
```

```
c ------------------------------------------------

      subroutine VarTriangle(nvar,pq)

c    Si les parametres sont  des nombres flous triangulaire , on incremente le tablea
u pq de 1 pour chaque parametre.
c
c     Entree
c     ------
c     nvar    : Le nombre de variable
c
c     Sortie
c     ------
c     pq    : Un tableau a une deux contenant toutes les p et q des fonctions L er R d
es parametres Ai
c     ---------------------------------------
      include 'info.ins'

      integer nvar,i
      real pq(2,nvar1)

      i=1
      do while (i.ne.nvar+2)
         pq(1,i)=1
         pq(2,i)=1
         i=i+1
      end do
      end


c ------------------------------------------------


      subroutine RemplirTab(tab,obs,nvar,nobs,h,pq,yflou,moyenne)

c Les entrees:
c -----------
c     nobs      : le nombre d'observations
c     h         : La hauteur de la regression (son niveau)
c     yflou     : Le boolean verifiant si les y observes sont des nombres flous
c     obs       : Le tableau de donnees (d'observations)
c     nvar      : Le nombre de variables explicatives
c     pq        : Un tableau a une deux contenant toutes les p et q des fonctions L e
r R des parametres Ai
c     moyenne   : Un tableau a une dimensions contenant toutes les moyennes des varia
bles Xi
c
c Les resultats:
c -------------
c     tab       : La tableau contenant les contraines et la fonction a maximiserr pou
r le simplex.

      include 'info.ins'

      real tab(LargTab,LongTab),obs(nvar5,nobs0),pq(2,nvar1)
      real moyenne(nvar0),h
      integer yflou,nvar,nobs

      call FonctionMinimiser(tab,obs,nvar,nobs,pq,moyenne)
```

```
      call ContInf(tab,obs,nvar,nobs,h,pq,yflou,moyenne)

      call ContSup(tab,obs,nvar,nobs,h,pq,yflou,moyenne)
c      print *,' Apres Contraintes Sup..'
c      print '(32f10.2)', tab

      end



c --------------------------------------------------

      subroutine ContSup(tab,obs,nvar,nobs,h,pq,yflou,moyenne)

c Les entrees:
c -----------
c     nobs       : le nombre d'observations
c     h          : La hauteur de la regression (son niveau)
c     yflou      : Le boolean verifiant si les y observes sont des nombres flous
c     obs        : Le tableau de donnees (d'observations)
c     nvar       : Le nombre de variables explicatives
c     pq         : Un tableau a une deux contenant toutes les p et q des fonctions L e
r R des parametres Ai
c     moyenne    : Un tableau a une dimensions contenant toutes les moyennes des varia
bles Xi
c
c Les resultats:
c -------------
c     tab        : La tableau contenant les contraines et la fonction a maximiser pour
 le simplex.
      include 'info.ins'

      real tab(LargTab,LongTab),obs(nvar5,nobs0),pq(2,nvar1)
      real moyenne(nvar0),h,max,par
      integer yflou,i,j,k,nvar,nobs,param2


      i=nobs+2
      do while (i.ne.2*nobs+2)
         param2=i-(nobs+1)
c     Calcul de la valeur max du nombre fou Yi observe ,
c     pour le niveau h, renvoi Yi si celui ci n'est pas un nombre flou
         call calmax(obs,yflou,nvar,max,param2,h)
         tab(i,1)=max
         tab(i,2)=-1
         tab(i,3)=0
         tab(i,4)=-(1-h)**(1/pq(2,1))
         j=1
         k=5
         do while (j.ne.nvar+1)
c     Calcul de par: la difference entre la velur de la variable observee et sa moyen
ne
            par = obs(j,i-(nobs+1))- moyenne(j)
            if (par.ge.0) then
               tab(i,k)=-par
               tab(i,k+1)=0
               tab(i,k+2)=-par*(1-h)**(1/pq(2,j+1))
            else
               tab(i,k)=-par
               tab(i,k+1)=par*(1-h)**(1/pq(1,j+1))
```

```
               tab(i,k+2)=0
            end if
            j=j+1
            k=k+3
         end do
         i=i+1

      end do
      end


c -------------------------------------------------
      subroutine ContInf(tab,obs,nvar,nobs,h,pq,yflou,moyenne)

c Les entrees:
c -----------
c     nobs       : le nombre d'observations
c     h          : La hauteur de la regression (son niveau)
c     yflou      : Le boolean verifiant si les y observes sont des nombres flous
c     obs        : Le tableau de donnees (d'observations)
c     nvar       : Le nombre de variables explicatives
c     pq         : Un tableau a une deux contenant toutes les p et q des fonctions L e
r R des parametres Ai
c     moyenne    : Un tableau a une dimensions contenant toutes les moyennes des varia
bles Xi
c
c Les resultats:
c -------------
c     tab        : La tableau contenant les contraines et la fonction a maximiserr pou
r le simplex.

      include 'info.ins'

      real tab(LargTab,LongTab),obs(nvar5,nobs0),pq(2,nvar1)
      real moyenne(nvar0),h,min,par
      integer yflou,i,j,k,nvar,nobs,param2

      print *, 'Rentre dans continf'
      i=2
      do while (i.ne.nobs+2)
         param2=i-1
c     Calcul de la valeur min du nombre fou Yi observe ,
c     pour le niveau h, renvoi Yi si celui ci n'est pas un nombre flou
         call calMin(obs,yflou,nvar,min,param2,h)
         tab(i,1)=min
         tab(i,2)=-1
         tab(i,3)=(1-h)**(1/pq(1,1))
         tab(i,4)=0
         j=1
         k=5
         do while (j.ne.nvar+1)
c     Calcul de par: la difference entre la velur de la variable observee et sa moyen
ne
            par = obs(j,i-1)- moyenne(j)
            if (par.ge.0) then
               tab(i,k)=-par
               tab(i,k+1)=par*(1-h)**(1/pq(1,j+1))
               tab(i,k+2)=0
            else
               tab(i,k)=-par
```

```
         par=(di**2+ei**2)/(2*(di+ei))
         tab(1,3*i+2)=0
         tab(1,3*i+3)=-par*(1-((1**(pq(1,i)+1))/(pq(1,i)+1)))
         tab(1,3*i+4)=-par*(1-((1**(pq(2,i)+1))/(pq(2,i)+1)))
c         print *, '  i:',3*i,'   ',tab(1,3*i+3), '   ',tab(1,3*i+4)
         i=i+1
      end do


      end



c -------------------------------------------------
      subroutine calMax(obs,yflou,nvar,max,j,h)

c Les entrees:
c -----------
c     obs       : Le tableau de donnees (d'observations)
c     nvar      : Le nombre de variables explicatives
c     h         : La hauteur de la regression (son niveau)
c     yflou     : Le boolean verifiant si les y observes sont des nombres flous
c     j         : l'indice de l'observation utilisee
c Les resultats:
c -------------
c     max       : La valeur maximum de la variable observee Yi
c                 pour la j eme observation et au niveau h

      include 'info.ins'

      real obs(nvar5,nobs0),max
      integer yflou,nvar,j,h

      print*, ' calcul max de ', j
      if (yflou.ne.1) then
         max=obs(nvar+1,j)
      else
         max=obs(nvar+1,j)+(obs(nvar+3,j)*((1-h)**(1/obs(nvar+5,j))))
      end if
      print *,' le max est : ',max
      end


c -------------------------------------------------
      subroutine calMin(obs,yflou,nvar,min,j,h)

c Les entrees:
c -----------
c     obs       : Le tableau de donnees (d'observations)
c     nvar      : Le nombre de variables explicatives
c     h         : La hauteur de la regression (son niveau)
c     yflou     : Le boolean verifiant si les y observes sont des nombres flous
c     j         : l'indice de l'observation utilisee
c Les resultats:
c -------------
c     min       : La valeur minimum de la variable observee Yi
c                 pour la j eme observation et au niveau h

      include 'info.ins'
```

```
      real obs(nvar5,nobs0),min,h
      integer yflou,nvar,j

      if (yflou.ne.1) then
         min=obs(nvar+1,j)
         print *, 'min    ', min
      else
         min=obs(nvar+1,j)-(obs(nvar+2,j)*((1-h)**(1/obs(nvar+4,j))))
      end if
      end
```

```
      SUBROUTINE simplx(a,m,n,mp,np,m1,m2,m3,icase,izrov,iposv)

      include 'info.ins'

      INTEGER icase,m,m1,m2,m3,mp,n,np,iposv(m),izrov(n),MMAX,NMAX
      REAL a(mp,np),EPS
      PARAMETER (MMAX=100,NMAX=100,EPS=1.e-2)
CU    USES simp1,simp2,simp3
      INTEGER i,ip,is,k,kh,kp,nl1,l1(NMAX),l3(MMAX)
      REAL bmax,q1
      if(m.ne.m1+m2+m3)pause 'bad input constraint counts in simplx'
      nl1=n
      do 11 k=1,n
         l1(k)=k
         izrov(k)=k
 11   enddo
      do 12 i=1,m
         if(a(i+1,1).lt.0.)pause 'bad input tableau in simplx'
         iposv(i)=n+i
 12   enddo
      if(m2+m3.eq.0)goto 30
      do 13 i=1,m2
         l3(i)=1
 13   enddo
      do 15 k=1,n+1
         q1=0.
         do 14 i=m1+1,m
            q1=q1+a(i+1,k)
 14      enddo
         a(m+2,k)=-q1
 15   enddo
      print *, ' Apres la somme..'
      print  '(32f18.2)', a
 10   call simp1(a,mp,np,m+1,l1,nl1,0,kp,bmax)
      if(bmax.le.EPS.and.a(m+2,1).lt.-EPS)then
         print *, ' Pass 1'
         icase=-1
         return
      else if(bmax.le.EPS.and.a(m+2,1).le.EPS)then
         do 16 ip=m1+m2+1,m
            if(iposv(ip).eq.ip+n)then
               call simp1(a,mp,np,ip,l1,nl1,1,kp,bmax)
               if(bmax.gt.EPS)goto 1
            endif
 16      enddo
         do 18 i=m1+1,m1+m2
            if(l3(i-m1).eq.1)then
               do 17 k=1,n+1
                  a(i+1,k)=-a(i+1,k)
 17            enddo
            endif
 18      enddo
         goto 30
      endif
      call simp2(a,m,n,mp,np,ip,kp)
      if(ip.eq.0)then
         print *, ' Pass 2'
         icase=-1
         return
      endif
```

```
 1    call simp3(a,mp,np,m+1,n,ip,kp)
      if(iposv(ip).ge.n+m1+m2+1)then
         do 19 k=1,nl1
            if(l1(k).eq.kp)goto 2
 19      enddo
 2       nl1=nl1-1
         do 21 is=k,nl1
            l1(is)=l1(is+1)
 21      enddo
      else
         kh=iposv(ip)-m1-n
         if(kh.ge.1) then
            if (l3(kh).ne.0) then
               l3(kh)=0
               a(m+2,kp+1)=a(m+2,kp+1)+1.
               do 22 i=1,m+2
                  a(i,kp+1)=-a(i,kp+1)
 22            enddo
            endif
         endif
      endif
      is=izrov(kp)
      izrov(kp)=iposv(ip)
      iposv(ip)=is
      goto 10

 30   call simp1(a,mp,np,0,l1,nl1,0,kp,bmax)
      if(bmax.le.EPS)then
         icase=0
         return
      endif
      call simp2(a,m,n,mp,np,ip,kp)
      if(ip.eq.0)then
         icase=1
         return
      endif
      call simp3(a,mp,np,m,n,ip,kp)
      is=izrov(kp)
      izrov(kp)=iposv(ip)
      iposv(ip)=is
      goto 30
      END



C     (C) Copr. 1986-92 Numerical Recipes Software .
      SUBROUTINE simp1(a,mp,np,mm,ll,nll,iabf,kp,bmax)
      INTEGER iabf,kp,mm,mp,nll,np,ll(np)
      REAL bmax,a(mp,np)
      INTEGER k
      REAL test


      print *, 'ON RENTRE DANS SIMP1 '
      if(nll.le.0) then
         bmax=0
      else
         kp=ll(1)
         bmax=a(mm+1,kp+1)
         print *, 'bmax: ' ,bmax
```

```
      do 11 k=2,nll
         if(iabf.eq.0)then
            test=a(mm+1,ll(k)+1)-bmax
         else
            test=abs(a(mm+1,ll(k)+1))-abs(bmax)
         endif
         if(test.gt.0.)then
            bmax=a(mm+1,ll(k)+1)
            kp=ll(k)
         endif
11    enddo
   endif
   return
   END


C     (C) Copr. 1986-92 Numerical Recipes Software .
   SUBROUTINE simp2(a,m,n,mp,np,ip,kp)
   INTEGER ip,kp,m,mp,n,np
   REAL a(mp,np),EPS
   PARAMETER (EPS=1.e-2)
   INTEGER i,k
   REAL q,q0,q1,qp

   print *, ' ON RENTRE DANS SIMP2'
   ip=0
   do 11 i=1,m
      print *, ' i : ' ,i,' kp  ', kp
      print *, a(i+1,kp+1)
      if(a(i+1,kp+1).lt.-EPS) then
         print *, '< EPS '
         goto 1
      endif
11 enddo
   print *, ' ip=  ', ip
   return
1  q1=-a(i+1,1)/a(i+1,kp+1)
   ip=i
   do 13 i=ip+1,m
      if(a(i+1,kp+1).lt.-EPS)then
         q=-a(i+1,1)/a(i+1,kp+1)
         if(q.lt.q1)then
            ip=i
            q1=q
         else if (q.eq.q1) then
            do 12 k=1,n
               qp=-a(ip+1,k+1)/a(ip+1,kp+1)
               q0=-a(i+1,k+1)/a(i+1,kp+1)
               if(q0.ne.qp)goto 2
12          enddo
2           if(q0.lt.qp)ip=i
         endif
      endif
13 enddo
   print *, ' ip=  ', ip
   return
   END


C     (C) Copr. 1986-92 Numerical Recipes Software .
```

```
      SUBROUTINE simp3(a,mp,np,i1,k1,ip,kp)
      INTEGER i1,ip,k1,kp,mp,np
      REAL a(mp,np)
      INTEGER ii,kk
      REAL piv

      print *, 'ON RENTRE DANS SIMP3'
      piv=1./a(ip+1,kp+1)
      do 12 ii=1,i1+1
        if(ii-1.ne.ip)then
           a(ii,kp+1)=a(ii,kp+1)*piv
           do 11 kk=1,k1+1
            if(kk-1.ne.kp)then
               a(ii,kk)=a(ii,kk)-a(ip+1,kk)*a(ii,kp+1)
            endif
 11       enddo
        endif
 12   enddo
      do 13 kk=1,k1+1
         if(kk-1.ne.kp)a(ip+1,kk)=-a(ip+1,kk)*piv
 13   enddo
      a(ip+1,kp+1)=piv
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software .
```

```
               tab(i,k+1)=0
               tab(i,k+2)=-par*(1-h)**(1/pq(1,j+1))
            end if
            j=j+1
            k=k+3
         end do
         i=i+1

      end do

      end
c -------------------------------------------------

      subroutine FonctionMinimiser(tab,obs,nvar,nobs,pq,moyenne)

c Les entrees:
c -----------
c     nobs      : le nombre d'observations
c     obs       : Le tableau de donnees (d'observations)
c     nvar      : Le nombre de variables explicatives
c     pq        : Un tableau a une deux contenant toutes les p et q des fonctions L e
r R des parametres Ai
c     moyenne   : Un tableau a une dimensions contenant toutes les moyennes des varia
bles Xi
c
c Les resultats:
c -------------
c     tab       : La tableau contenant les contraines et la fonction a maximiserr pou
r le simplex.
      include 'info.ins'

      real tab(LargTab,LongTab),obs(nvar5,nobs0),pq(2,nvar1),ei,di
      real moyenne(nvar0),min,max,par
      integer nvar,nobs,i,j

      tab(1,1)=0
      tab(1,2)=0
      tab(1,3)=-(1-((1**(pq(1,1)+1))/(pq(1,1)+1)))
      tab(1,4)=-(1-((1**(pq(2,1)+1))/(pq(2,1)+1)))
      i=1
c     ------------------------------------
c     Calcul du minimum et du maximun de chaque variabe explicative Xi

      do while (i.ne.nvar+1)
         min=obs(i,1)
         max=obs(i,1)
         j=2
         do while(j.ne.nobs+1)
            if (obs(i,j).lt.min) then
               min=obs(i,j)
            else
               if (obs(i,j).gt.max) then
                  max=obs(i,j)
               end if
            end if
            j=j+1
         end do
c     ------------------------------------
         ei=max-moyenne(i)
         di=moyenne(i)-min
```

```
      subroutine resultsimplex(tab,icase,izrov,iposv,Ai,npar1,
     &     npar2,pq,nvar)


      include 'info.ins'

      integer npar1,npar2,nvar
      integer icase,izrov(LongTab),iposv(LargTab),i,j
      real tab(LargTab,LongTab),Ai(5,nvar1),pq(2,nvar1)


      print *,' entre dans resultat'
c     Test sur la validite du resultat

      if (icase.ne.0) then
         return
      end if

      i=1
c     Parcours du tableu izrov et affecte dans Ai les resultats du simplex
      do while(i.ne.npar2+2)
         if (izrov(i).le.npar2) then
            if ( mod(izrov(i),3).ne.0) then
               Ai( mod(izrov(i),3),(izrov(i)/3) +1)=tab(1,i+1)
            else
               Ai(3,(izrov(i)/3))=tab(1,i+1)
            end if
         end if
         i=i+1
      end do

      j=1
c     Parcours du tableu iposv et affecte dans Ai les resultats du simplex
      do while(j.ne.npar1+2)
         if (iposv(j).le.npar2) then
            if ( mod(iposv(j),3).ne.0) then
               Ai( mod(iposv(j),3),(iposv(j)/3)+1)=tab(j+1,1)
            else
               Ai (3,(iposv(j)/3))=tab(j+1,1)
            end if
         end if
         j=j+1
      end do

c     Affecte les p q dans le tableau Ai
      i=1
      do while(i.ne.nvar+2)
         Ai(4,i)=pq(i,1)
         Ai(5,i)=pq(i,2)
         i=i+1
      end do
      end



      subroutine estimation(nobs,nvar,obs,Ai,deflou,estim,yflou)


      include 'info.ins'
```

```
      real obs(nvar5,nobs0),Ai(5,nvar1),estim(nobs0)
      integer nobs,nvar,deflou,yflou

      real*8    a1(5),a2(5)
      real*4    x1(np0),y1(np0),x2(np0),y2(np0),x(np0),y(np0),z
      integer*4 itype
      integer*4 np,np1,np2
      integer*4 ierr
      integer i,nj,k


      print *,' entre dans estimation'
      itype=2
      nj=1

c     Pour chaque observation valcul des yi estimes
      do while(nj.ne.nobs+1)

c     Incrementation du tableau a1 pour l'utilisation de Modflou
         a1(2)=Ai(1,1)
         if (Ai(2,1).le.0) then
            a1(1)=a1(2)
         else
            a1(1)=a1(2)-Ai(2,1)
         endif
         if (Ai(3,1).le.0) then
            a1(3)=a1(2)
         else
            a1(3)=a1(2)+Ai(3,1)
         endif

         a1(4)=Ai(4,1)
         a1(5)=Ai(5,1)
         print *, ' A1  ',a1
c     Discretisation de a1
         call fnumb2xy_c(itype,a1,np0,np1,x1,y1,ierr)

         i=2
         do while(i.ne.nvar+2)

c     Incrementation du tableau a2 pour l'utilisation de Modflou
            a2(2)=Ai(1,i)*obs(i-1,nj)
            if (Ai(2,i).le.0) then
               a2(1)=a2(2)
            else
               a2(1)=a2(2)-(Ai(2,i)*obs(i-1,nj))
            endif
            if (Ai(3,i).le.0) then
               a2(3)=a2(2)
            else
               a2(3)=a2(2)+(Ai(3,i)*obs(i-1,nj))
            endif
            a2(4)=Ai(4,i)
            a2(5)=Ai(5,i)

            print*,' A2 ', a2
c     Discretisation de a2
            call fnumb2xy_c(itype,a2,np0,np2,x2,y2,ierr)
c     Somme de a1 et a2
```

```
            call fsetop_c('+',np1,x1,y1,np2,x2,y2,np,x,y,ierr)
            k=1
c     Affectation de a1:  a1=a1+a2
            do while(k.ne.np+1)
               x1(k)=x(k)
               y1(k)=y(k)
               k=k+1
            enddo
            np1=np
            i=i+1
         enddo
c     Une fois les calculs effectues pour toutes les variables, deflouification
         call  deflou_c(deflou,np1,x1,y1,z,ierr)
c     Affichage du yi estime et de l'observe
         print *,' Z ',z
         estim(nj)=z
         print *, ' ESTIMATION DE NJ= ',nj, '     ', estim(nj)
         nj=nj+1
      enddo
      print *,estim
      end




      subroutine Resultat(Ai,estim,nobs,nvar,mode,ficflou,ficresult)

      include 'info.ins'

      character*128 ficflou,ficresult
      real estim(nobs0),Ai(5,nvar1)
      integer mode,i,j,nobs,nvar

      if (mode.eq.'cal') then
         open (1,FILE=ficresult,ACCESS='Direct',RECL=11,STATUS='new')
         i=1
         do while (i.ne.nobs+1)
            write(1,REc=i) estim(i)
            i=i+1
         end do
      end if
      open (1,FILE=ficflou,ACCESS='Direct',RECL=11,STATUS='new')
      i=1
      do while (i.ne.nvar+2)
         write(1,REc=i) (Ai(i,j),j=1,5)
         i=i+1
      enddo
      end
```